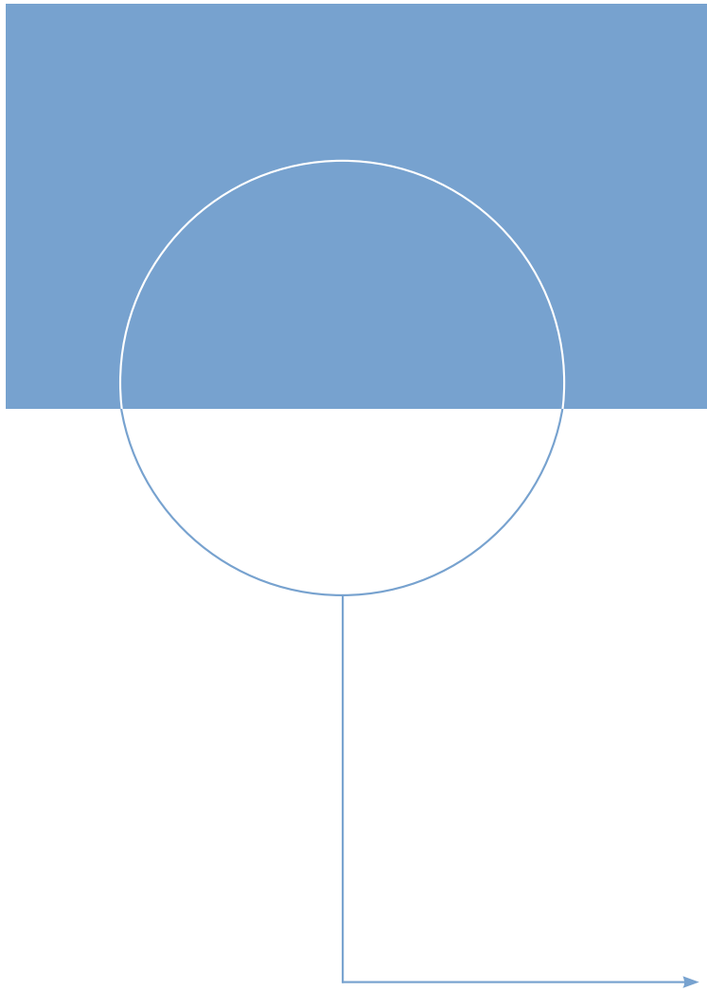

Master's thesis

2020

Magnus Hammer Zakariassen

**NTNU**
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Master's thesis

Magnus Hammer Zakariassen

# Automatic pellet detection with Computer Vision in aquaculture

August 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

# Automatic pellet detection with Computer Vision in aquaculture

Magnus Hammer Zakariassen

Cybernetics and Robotics
Submission date: August 2020
Supervisor: Morten Alver

Norwegian University of Science and Technology
Department of Engineering Cybernetics

# Abstract

The waste of fish food has is a serious problem in aquaculture. Any uneaten fish food contributes to economic loss in the industry, as costs related to food accounts for roughly 40-50% of the total production costs in fish farming. This project researches the possibility of aiding in the automation of the feeding process through automatically detecting pellets leveraging computer vision and deep learning technology.

The solution utilizes a convolutional neural network, which is considerer to be state-of-the-art within object detection. Models based on convolutional neural networks have delivered respectable results within classification and detection of object, and is therefore a natural choice for this research.

The final model is currently not accurate enough for deployment in an industrial environment. However, the model shows signs potential in certain scenarios, namely when the pellets are in close proximity to the camera and has good visibility. From the end result it is apparent that further research with a larger, more robust dataset is necessary to improve the model's performance.

# Sammendrag

Matsvinn er et problem i oppdrettsnæring. Fôr som forblir uspist bidrar til store økonomiske tap i industrien, ettersom kostnader relatert til fôr utgjør 40-50% av totale produksjonskostnader. Denne oppgaven undersøker om maskinsyn basert på dyp læring kan bidra til å automatisere hele eller deler av fôringsprosessen ved å automatisk detektere synkende pellets.

Løsningen benytter et konvolusjonelt nevralt nettverk, som regnes som state-of-the-art innen objektdeteksjon. Modeller basert på konvolusjonelle nevrale nettverk har levert gode resultater på klassifisering og detektering av objekter, og ses derfor på som et naturlig valg.

Modellen er foreløpig ikke robust nok til å helautomatisere oppdrettsanlegg, men viser lovende resultater i gitte situasjoner. Eksempelvis når pelletsene er i nærheten av kameraet og det er relativt uhindret sikt. Ut i fra resultatene er det tydelig at et større og mer robust datasett er nødvendig for å forbedre modellens ytelse.

# Preface

This thesis concludes the two year Master's program within Cybernetics and Robotics and the Norwegian University of Science and Technology. It was written during the late spring and summer of 2020 in collaboration with Piscada.

I wish to thank Piscada for giving me the opportunity to work on this project. I would particularly like to thank Olav Jamtøy and Knut Drange from Piscada for granting me access the video footage used to create the dataset. My supervisor from NTNU, Morten Alver, also for guidance and constructive feedback during this process.

# Table of Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

## 1.1 Motivation

Aquaculture and worldwide export of fish is Norways second largest export industry, only beaten by the oil  gas sector. In 2019 Norway exported 1.2 million tonnes of fish from aquaculture for a total value of 76.5 billion NOK and further growth is expected in the coming years [31, 32]. Feed represents about 40-50 % of the production cost [7, 33].

In the salmon fish farming industry the feeding process is controlled manually by an operator. From controlling and watching live underwater video, the operator makes experience-based decisions on when to manually adjust the feeding. The decisions are based on a combination of the behaviour of the fish and if a significant amount of fish feed sinks past the first few meters below surface. Both of

The average fish per farm saw a rise from 450 000 to 600 000 individual fish per site in the time frame from 2006-2009, which we can assume has grown since then given the huge growth in fish from aquaculture[16]. As the industry continues to scale up it will greatly benefit from the economics of scale, especially as sub-processes are partially or fully automated. There are both economical and environmental interest in improving the efficiency through automation of the feeding process in aquaculture.

## 1.2 Introduction of the project

Piscada, in collaboration with Norwegian Fish Farms, seeks to automate parts of the feeding process in aquatic farming of salmon. An important role of the operator is to determine whether the fish is being under- or overfed, and adjust the feeding supply accordingly. From a control theory perspective, the operator is estimating the state, error and providing the feedback manually to adjust the input. The first step in improving the efficiency of this process is to aid the decision making for the operator judging whether the fish is being

under- or overfed. The intention is to use an object detection algorithm, utilizing the same camera technology operators are using, for detecting pellets.

## 1.3 Project aim and objectives

One of the major obstacles for using computer vision in a fish farming enviroment are the multiple, continuously changing noise factors such as sunlight and chlorophyll from algae [15, 40]. Over the span of a year the visibility will have significant fluctuations. There are also factors that affects how much the fish will eat such as water temperature, but they are not part of the project scope.

The project aims to:

- Research if a model based on deep learning learning can detect pellets in a fish farm
- Identify if certain data augmentation techniques significantly enhances model performance

.

Chapter 2

# Theory

## 2.1 Neural Networks

Neural networks are the most common form of applying supervised deep learning. The network consist of several components and mathematical principles that will be explained in this section.

### 2.1.1 Artificial neuron

An artificial neuron is a mathematical function mimicking biological neurons in the human brain. It was first described in 1943 [26], and has since laid the foundation for neural networks.

The simplest form of an artificial neuron is a perceptron [5]

$$f(x) = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases} \tag{2.1}$$

The neruons used in modern nerual networks are more sophisticated and works by summing up all weighted inputs $wa^{n-1}$, adding bias $b$ and passing it through an activation function $f$ with output $a^n$, where $n$ is the layer

$$y^n = f\Big(\sum Wy^{n-1} + b\Big) \tag{2.2}$$

Equation (2.2) is on vectorized form, and is explained on a single neuron in figure 2.1, where the activation function $f$(explained later) is applied on the input $z$ to get output $y = f(z)$. $x_1, \ldots, x_D$ represent input from other units within the networ,; $b_0$ is the bias and represents an external input to the unit [38].

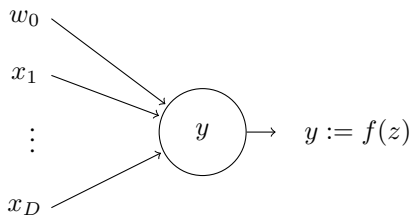


**Figure 2.1:** Single artificial neuron with components.

### 2.1.2 Fully connected network

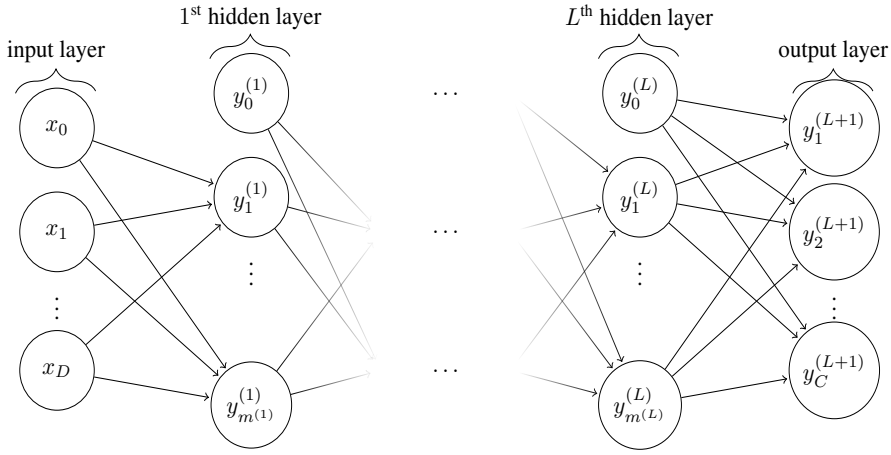


**Figure 2.2:** Neural network with $L + 1$ layers, $m^{(l)}$ hidden units, $D$ inputs and $C$ different classes

Figure 2.2 shows a fully connected neural network. The size of the input layer depends on the shape of the data, for example a network taking $100x100$ sized images as input will have 10 000 input units. The hidden units in each layer are a series of perceptrons or more sophisticated artifial neruons, taking a weighted input from all the nodes in the previous layer as explained earlier in this section. The amount of layers and hidden units affects the networks complexity, and although a network with more units can approximate more complex data, it also requires more data to train and has higher risk of overfitting [35].

### 2.1.3 Learning

The learning is composed of several processes. An input passes through a network of neurons as described earlier in figure 2.2. After passing through. The output is then compared to the label and the network is updated accordingly using backpropagation. The elements of the nework are explained in higher detail.

### Activation function

The purpose of an activation functions in the context neural networks is to introduce non-linearity between layers and for prediction at the output layer. A standard multilayer feed-forward network with a locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial [22]. Some common activation functions will be described.

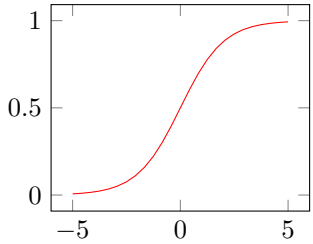


**Figure 2.3:** Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

The sigmoid function, described by equation (2.3), was used to numerically describe biological neurons' response to various stimulus in 1972 [36]. It has been utilized both in classic approaches to machine learning, such as SVM, and later in neural networks mainly as an activation function [23, 39].
As opposed to the primitive perceptron, which uses a simple step-function, the sigmoid has a smooth curve as seen in figure 2.3.

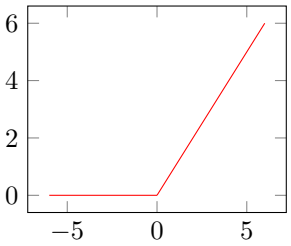


**Figure 2.4:** Rectified linear unit (ReLU)

$$f(x) = max(0, x) \tag{2.4}$$

The Rectified Linear Unit (ReLU) described by equation(2.4), resembles the perceptron in terms of having a threshold for activation as seen in figure 2.4. ReLU has several qualities and outperforms the Sigmoid both in terms of convergence accuracy and convergence rate [11] become the most popular activation function for use between layers [1].

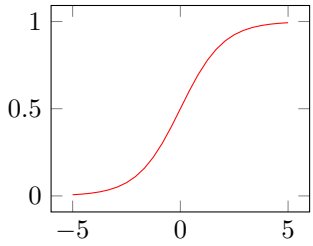


**Figure 2.5:** Softmax

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.5}$$

The softmax function (2.5) looks strikingly similar to the sigmoid as seen in figure 2.5. However, it has a different purpose in a neural network. The softmax squeezes the values of a $K$ dimentional vector $z$ into a $K$ dimentional vector with real values between 0 and 1. The sum of these values are equal to 1. This is used in the last layer to classify objects, and each element $j$ of the vector represents the model's output probability of the corresponding class [17].

For example if we have a model that classifies the animals dog, cat and horse, feed it with an input image and get the output vector $[0.3, 0.65, 0.05]^T$ after the softmax, the model predicted that it is a 65% chance of being a cat.

### Backpropagation

In the field of deep learning, backpropagation is an algorithm used in training neural networks for supervised learning [29]. When optimizing a neural network to training data, often referred to as *fitting*, backpropagation computes the gradient of the loss function with respect to the weights of the network for every input–output example posed in the dataset. Unlike a direct method for computing the gradient with respect to each weight individually, which is slow and computationally expensive. Backpropagation computes the gradient of the loss function by utilizing the chain rule, thus iteratively computes the gradient backwards in the network, one layer at the time as seen in figure 2.6. This effectively avoids reduntant calculations, and as a result tremendously reduces computational cost, especially for larger network structures [13].

This computational efficiency makes it feasible to use gradient methods for updating weights to minimize loss when training multilayer networks. gradient descent, or variants such as stochastic gradient descent, are commonly used [6].

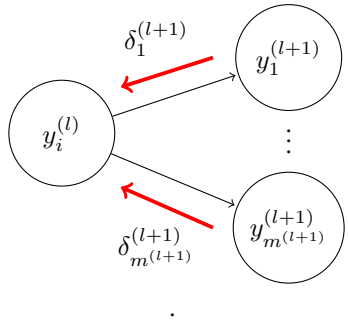


**Figure 2.6:** Backpropagation - the errors $\delta_i^{(L+1)}$ can be propagated backwards.

## 2.2 Convolution Neural Networks [CNNs]

A convolutional neural network learns and extracts features by utilizing convolutional filters and non-linear layers. This composition is referred to as a *feature exctraction subnet*, and as the name suggest detects features later used for classification. The feature extraction subnet can contain several convolution layers, each with multiple filters [18, 20, 21].

Figure 2.7 shows the arcitechture of the original CNN introduced by LeCun [21].

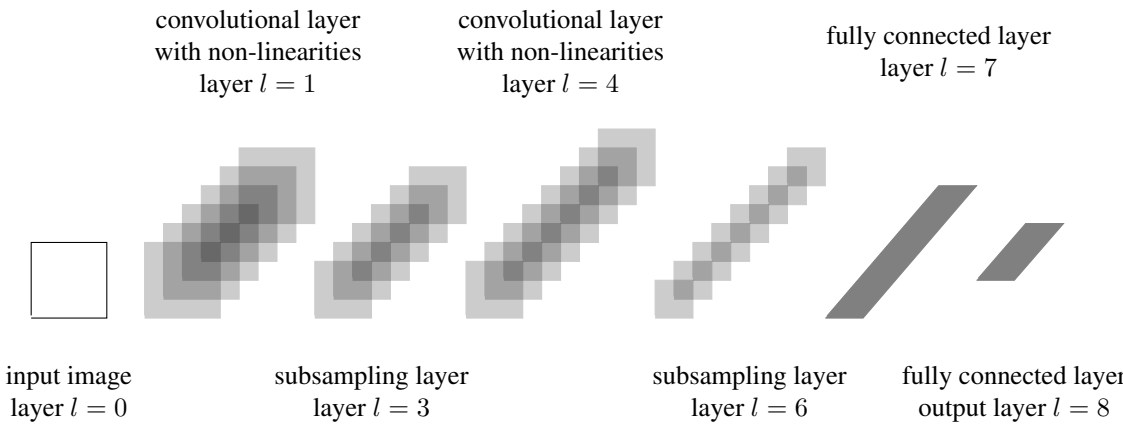


**Figure 2.7:** Original CNN as described by Lecun

### 2.2.1 Convolutional layer

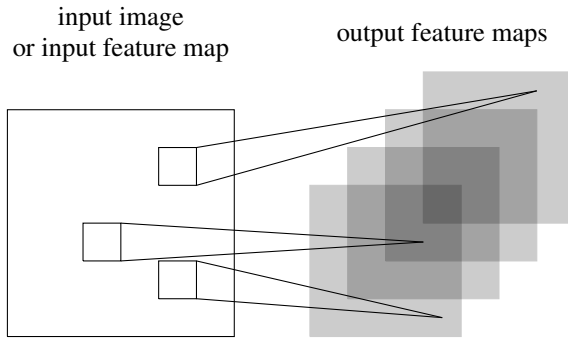


**Figure 2.8:** Single convolutional layer.

Figure 2.8 shows how a single convultional layer works, the original input is sent into the first convulutional layer, and subsequent layers uses the output feature map from the previous layer as its input [21].

### 2.2.2 Pooling layer

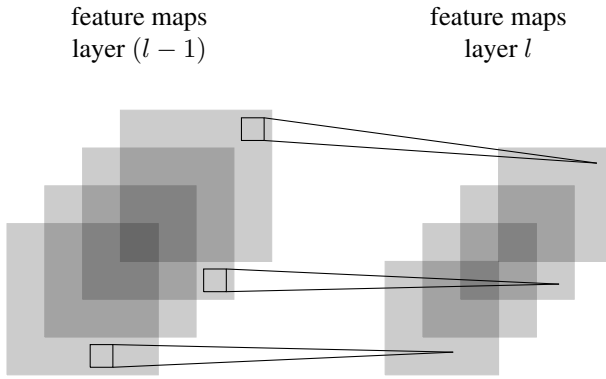


**Figure 2.9:** Illustration of a pooling and subsampling layer.

Figure 2.9 Shows how subsampling and pooling works in-between convolutional layers. The most common pooling type is max pooling, which takes the highest value of each $2x2$ sub-matrix as a single value in the output feature map.

### 2.2.3 Fully connected layer

A fully connected layer has every node connected to every node of the previous layer. In convolution neural networks it is often the last layer used for classification[21].

Chapter 3

# Related work

## 3.1 Deep learning

Machine learning and especially supervised learning, have made significant advancements the past decade. In 2012 when the first neural network based model won the ImageNet competition [19]. Research on supervised learning have since increased after it's proven potential, and in 2015 a neural network model beat human performance in the same competition[30].

### 3.1.1 Computer vision

Traditionally, the field of computer vision have used hand crafted features for object detection and classification. The main approaches have been Scale-invariant feature transform *(SIFT)* [25] and textures [12], along with statistical classifiers such as Support Vector Machines *(SVM)* [8] and Nearest Neighbour *(NN)* [3].
t
During recent years, more specifically since 2012 [18], deep learning methods have proven they can obtain higher accuracy compared to more traditional methods [28]. This trend have been enabled mainly by the improvement in hardware, such as GPUs, resulting in a significant increase in available computational power, leading to a surge in deep learning related technology [2]. Another important factor is the availability of large, labeled datasets such as Microsoft Common Objects in Context *(MS Coco)* [24] and ImageNet [9] containing more than 2.5 and 14 million images respectively.

The deep learning approach that have achieved the best results within computer vision, namely classification and object detection, is the Convolutional Neural Network *(CNN)*

### 3.1.2 Data augmentation

A rough rule of thumb is that supervised deep learning methods, such as SSD-MobileNet and Faster-RCNN, generally requires 5000 labeled images for every class to achieve acceptable performance [4, 10].

Has showed that by using images of the object you want to classify, performing transoformations on the object and applying random background, it is possible to create a synthetic dataset that delivers good results [27]

Chapter 4

# Method

## 4.1 Data

### 4.1.1 Collecting

The raw data is gathered from footage of various Norwegian fish farms using underwater cameras during the feeding process. Furthermore the images used in the dataset are sampled at a 3 second interval.

### 4.1.2 Cleanup

The dataset is cleaned up by manually removing outliers from the sampled data. Outliers are defined as images without fish feed.

### 4.1.3 Annotation

Data annotation is the process of labelling the data. The combination of a data point and a label makes up an input-output pair in the data set, essentially meaning that for a given input we expect the output from the model to be equal to the label. There are many different labelling tools available for annotation, both open source and closed source. For this project makesence.ai is used for annotating images. It is open source and runs locally through a browser. The software, along with other available open source software, has support for semi-automating the annotation process. However, it predominantly suggested annotation for the fish in the picture and struggled to identify the fish feed as objects. It therefore added more problems than it solved and was used.

### 4.1.4 Preprocessing

## 4.2 Training

The training uses several different types of data augmentation. Each training uses 120 000 epochs, referred to as *steps*, in the API.

## 4.3 Requirements

### 4.3.1 Hardware

The minimum requirement for running any form of machine/deep learning project is a computer with a CPU. As deep learning is computationally expensive, it is recommended to either use a GPU or a Tensor Processing unit *(TPU)* as they will significantly increase the training speed compared to a CPU [34]. Note that even though the GPU/TPU will significantly speed up the process, a weak or low-core CPU can similary be a bottleneck. It is desired to use a powerful multi-core CPU, as it is utilized in the pre-processing and communication with the GPU/TPU [37]. The exact specifications used in this project are stated in table 4.1.

| Part | Specification |
|---|---|
| Processor | Intel Core i7-8700 |
| GPU | ASUS GeForce GTX 1080 |
| RAM | Ballistix Sport LT DDR4 16GB |
| Storage | Seagate Barracuda 1TB |
| PSU | Corsair TX550M, 550W |

**Table 4.1:** Hardware specifications

### 4.3.2 Software

### 4.3.3 Python

Python is a high-level programming language commonly used for deep learning projects. This project uses version 3.7.

#### Tensorflow Object Detection API

Tensorflow's Object Detection API is an open source framework for training and using deep learning architecture. The project uses version 1.15, as it at the time of this is the latest

version which support for training. The entire repository can be downloaded with git from https://github.com/tensorflow/models/tree/master/research/object$_detection$[14].$TofunctionproperlytheA$

| Package | Function | Version |
|---|---|---|
| Tensorflow-gpu | ML platform | 1.15 |
| Cudnn | GPU acceleration | 7.6.5 |
| Cudatoolkit | GPU acceleration | 10.0.130 |
| Pillow | Image manipulation | 1.0 |
| Lxml | Processing xml files | 4.5.0 |
| Protobuf | Serializing structured data | 3.0.0 |
| Cython | Python utilizing C/C++ | - |
| contextlib2 | Context manager | - |
| Matplotlib | Plotting/visualisation | - |

**Table 4.2:** Package and library requirements

Chapter 5

# Results

## 5.1 Statistical results

The statistical results are the average output from every trained model. The accuracy seen in table 5.1, we observe that the models using data augmented with flipping, jitter boxes and black patches all produce similar results. The models trained on data augmented by adjusting brigtness or HUE performed significantly worse.

| Augmentation type | Accuracy |
|---|---|
| Only flipped | 0.312 |
| Jitter boxes | 0.323 |
| Black patches | 0.314 |
| Adjust brightness | 0.181 |
| Adjust HUE | 0.156 |

**Table 5.1:** Total accuracy on the test set

The loss on the training set is not used for determining actual model performance, but to indicate convergence of the models. The total loss at the end of training can be seen in table 5.2

| Augmentation type | Total loss |
|---|---|
| Only flipped | 2.800 |
| Jitter boxes | 2.992 |
| Black patches | 3.028 |
| Adjust brightness | 8.542 |
| Adjust HUE | 8.517 |

**Table 5.2:** Total loss at the end of training

## 5.2 Empirical results

To give more context to the statistical performance metrics, there picked out a a few images from the test dataset to observe the model performance. As the accuracy is fairly similar on all the best performing models, the best model was used for the tested images.

From figure 5.1 and 5.2, we observe that the model detects the pellets with certainty in both images. Both images has a relatively unrestricted view with good visibility. Note that there is also a clear background behind the pellets and fairly little overlap with fish.

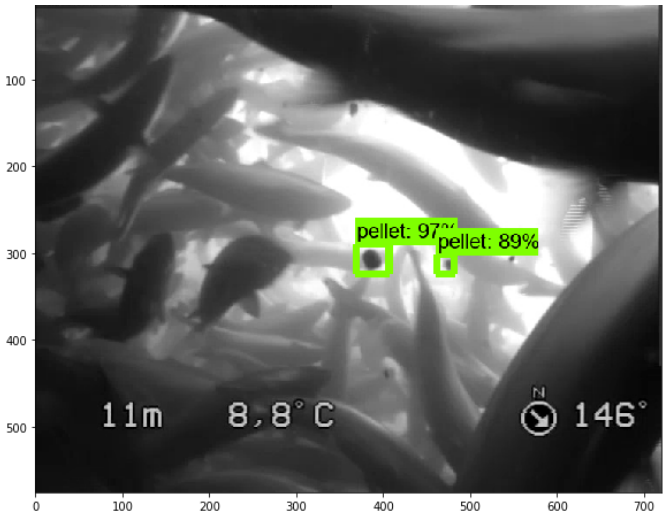


**Figure 5.1:** Model performance on image with high visibility to pellets

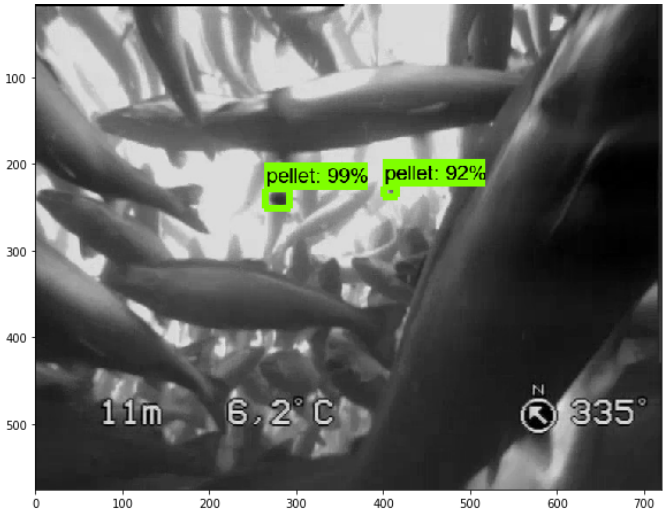


**Figure 5.2:** Model performance on image with high visibility to pellets

From figure 5.3 and 5.4, we observe that the model struggles to detect the pellets. In both images it is only able to detect 1, and also with significantly lower certainty compared to the detection from the images in figure 5.1 and 5.2. The pellets are much farther away from the camera, and while still being visible they are noticeably harder to see for the human eye.


**Figure 5.3:** Model performance on image with pellets far away

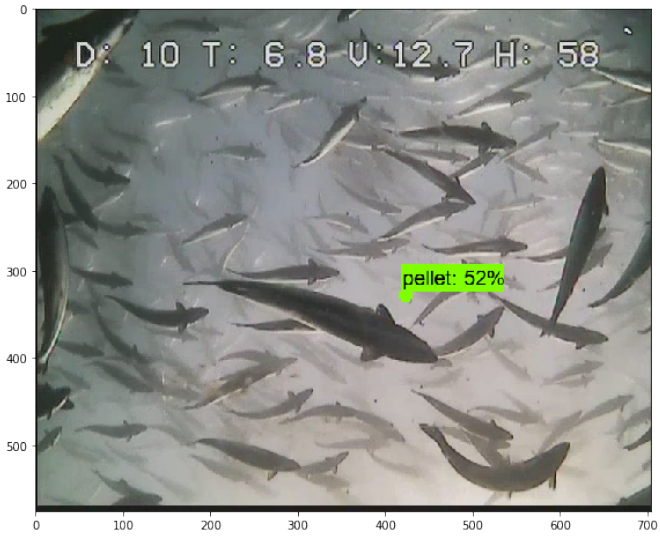


**Figure 5.4:** Model performance on image with pellets far away

# Chapter 6

# Discussion

From the empirical results we can observe that the model predicts pellets accurately and with high certainty on the images in figure 5.1 and 5.2. On these, and similar, pictures the fish feed takes up a relatively large portion of the picture and the camera has unrestricted view of the detected objects. Even though the images used for the empirical results are hand picked, and thus subject to some level of bias, the model indicated that it already have sufficiently high performance *in some situations*, which implies that the a convolutional neural network has potential to detect pellets.

However, the model struggles when the pellets further away as seen in figure 5.3 and 5.2. The overall accuracy of the model is therefore suffering and is currently not performing at a desired level to be used in production. There can be many reasons to the model not performing at a desired level. The model can be underfit due to training on a small dataset, or the model can be either too complex or lack complexity. As the SSD-Mobilenet have performed at high levels in other object detection tasks, it is likely not an issue related to model arcitechture. Considering that the data threshold for achieving high performance is roughly 5000 images [4, 10], the dataset used for this study can be considered fairly small.

## 6.1 Further work

### 6.1.1 Dataset

It is generally recommended to have a data set containing 5000 labeled images for every class the model should be able to detect. Given that the model used in production would operate in noisy environment with both short-term and long-term variance, it is also required to build a dataset that represents the noisy environment. Increasing the amount of available, labeled data would likely be the most important factor for improving the overall performance of the model. There are two main strategies to address this problem.

There are two main strategies to build a sufficiently large and diverse dataset. The trivial solution is to gather and label more data from video or pictures take, but it is not without downsides. Gathering and labelling data is both time/labour intensive and costly as well as prone to human error during the labelling process.

The second strategy is to expand the dataset artificially either through creating completely artificial data or through extensive augmentation of the current dataset. One method that have the potential to yield a large and diverse dataset, without requiring large amounts of raw data, is to create a two sub-datasets. One subset contains the object we want to detect, namely fish feed. The other subset consists of backgrounds. A large dataset can be created by picking one or more random samples from the first subset and inserting them with random size, orientation and position on a background from the second subset. To further enhance the diversity of the dataset, we can apply random data augmentation on both of the subsets before combining them.

### 6.1.2 Improved performance

Further studies on this topic should investigate if more advanced region proposal algorithms, such as state-of-the-art RCNN-based models, could improve performance for this task. Other network architectures, namely VGG and YOLO, can also be subject to further research.

### 6.1.3 Added parameters

By adding additional parameters to the model, such as weather, visibility in the water or other relevant information, it can aid in solving the problem of high variance noise.

### 6.1.4 Added functionality

If a model performs at a sufficiently high level to be deployed in production, it has many use cases. Adding extra functionality to aid in furhter autmation can be feasible.

#### Pellet density

Rather than detecting and counting pellets in the entire image, a system that calculates the pellet density over a given volume can be useful.

#### Mathematical model of biomass

Using an underlying mathematical model of the biomass in the farm, which for example describes the consumption rate of the fish and total amount of pellets already fed. This allows the input from the computer vision model to be weigthed, and prevents it from stopping or reducing the the

Chapter 7

# Conclusion

In this study the possibility of using a object detection model based on deep learning for use in aquaculture was researched. The overall accuracy of the model is currently not at a desirable level, but from the empirical analysis it was observed that the model shows high promise under certain circumstances, namely when the pellets is close to the camera and the visibility is high. With a limited raw dataset of 294 images. This suggests that a computer vision model based on deep learning have the potential to be a viable option in the aid of further automation in aquaculture. The study also looked at different data augmentation techniques, but has no significant findings or recommendations on using any type in particular. The author recommend that future research should focus on growing the dataset through collecting more raw data and using extensive data augmentation techniques.

# Bibliography


[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Charu C Aggarwal et al. *Neural networks and deep learning*. Springer, 2018.

[3] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[5] H. D. Block. The perceptron: A model for brain functioning. i. *Rev. Mod. Phys.*, 34:123–135, Jan 1962.

[6] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[7] CM Chang, W Fang, RC Jao, CZ Shyu, and IC Liao. Development of an intelligent feeding controller for indoor intensive culturing of eel. *Aquacultural engineering*, 32(2):343–353, 2005.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[12] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[13] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[14] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[15] Daniel M Jamu, Zhimin Lu, and Raul H Piedrahita. Relationship between secchi disk visibility and chlorophyll a in aquaculture ponds. *Aquaculture*, 170(3-4):205–214, 1999.

[16] Østen Jensen, T Dempster, EB Thorstad, I Uglem, and A Fredheim. Escapes of fishes from norwegian sea-cage aquaculture: causes, consequences and prevention. *Aquaculture Environment Interactions*, 1(1):71–83, 2010.

[17] AbdulWahab Kabani and Mahmoud R El-Sakka. Object detection and localization using deep convolutional networks with softmax activation and multi-class log loss. In *International Conference on Image Analysis and Recognition*, pages 358–366. Springer, 2016.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[20] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[21] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[22] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[23] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, 3(1-32):16, 2003.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[25] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[26] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[27] Daniel Mas Montserrat, Qian Lin, Jan Allebach, and Edward J Delp. Training object detection and recognition cnn models using data augmentation. *Electronic Imaging*, 2017(10):27–36, 2017.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278, 1994.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[31] Statistisk sentralbyrå. Akvakultur. 2019.

[32] Dag Sørli, Paul T Aandahl, and Ingrid Kristine Pettersen. Sjømateksport for 107,3 milliarder kroner i 2019. 2020.

[33] Ole Torrissen, Rolf Erik Olsen, Reidar Toresen, Gro Ingunn Hemre, Albert GJ Tacon, Frank Asche, Ronald W Hardy, and Santosh Lall. Atlantic salmon (salmo salar): the "super-chicken" of the sea? *Reviews in Fisheries Science*, 19(3):257–278, 2011.

[34] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

[35] Andreas Weigend. On overfitting and the effective number of hidden units. In *Proceedings of the 1993 connectionist models summer school*, volume 1, pages 335–342, 1994.

[36] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972.

[37] Wencong Xiao, Zhenhua Han, Hanyu Zhao, Xuan Peng, Quanlu Zhang, Fan Yang, and Lidong Zhou. Scheduling cpu for gpu-based deep learning jobs. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 503–503, 2018.

[38] RN Yadav, V Singh, and PK Kalra. Classification using single neuron. In *IEEE International Conference on Industrial Informatics, 2003. INDIN 2003. Proceedings.*, pages 124–129. IEEE, 2003.

[39] Mehdi Rezaeian Zadeh, Seifollah Amin, Davar Khalili, and Vijay P Singh. Daily outflow prediction by multi layer perceptron with logistic sigmoid and tangent sigmoid activation functions. *Water resources management*, 24(11):2673–2688, 2010.

[40] Boaz Zion. The use of computer vision technologies in aquaculture–a review. *Computers and electronics in agriculture*, 88:125–132, 2012.

# Appendix